by Allan E. Moose and
Marian J. Lorenz
~~156 Monell Avenue~~
~~Islip, New York 11751~~

Title: DISPLAY LISTS SIMPLIFIED

Blurb: Mix Modes on the Screen

An important step in understanding your ATARI's graphics capabilities is to create your own custom display lists. This article will show you, step-by-step, how to mix text and graphics on your TV screen. Our method uses BASIC commands to modify Graphics Modes 0 through 8. BASIC sacrifices some of the ATARI's flexibility; however, these techniques will help you eventually create display lists in Assembly Language.

The graphics capabilities of the ATARI are controlled by a microprocessor chip called ANTIC (Alpha-Numeric Television Interface Circuit). Any display list is a program for ANTIC.

There is a display list program provided automatically by each BASIC graphics command, or you can define your own. The display list specifies where screen data is located, what display modes to use, and any special display options ANTIC is to implement. Since the display list describes the screen from top to bottom, any mix of graphics or text modes can be displayed on the screen.

To understand displays, you need to know a bit about television. In a TV, a beam of electrons is shot at the screen. The beam starts at the top left-hand corner and moves across the screen. When it reaches the right-hand side, the beam is turned off, and returned to the left, and moved down slightly. It is then turned on again, and the process is repeated 262 times to form a completed screen image. When the beam reaches the bottom right-hand corner of the screen, it is turned off and returned to the top left-hand corner to start over. These horizontal sweeps are called scan lines and are the basis of the display. The scan-line pattern actually starts above and ends below the physical boundaries of the TV screen. To assure that no information is not displayed where you can't see it, the ATARI display is restricted to 192 scan lines, usually positioned in the middle of the screen.

FP1

DO NOT SET

```
 REM PROGRAM 1
 ? CHR$(125):GRAPHICS 0
0 DL=PEEK(560)+PEEK(561)*256:POKE DL+16,130
0 FOR J=0 TO 10:READ B:POKE 1536+J,B:NEXT J
5 DATA 72,169,42,141,10,212,141,24,208,104,64
0 POKE 512,0:POKE 513,6
5 POKE 54286,192
0 GOTO 40
```

DON'T SET

```
1 REM PROGRAM 2
5 GRAPHICS 0
10 DL=PEEK(560)+PEEK(561)*256:POKE DL+16,130
20 FOR J=0 TO 28:READ B:POKE 1536+J,B:NEXT J
25 DATA 72,138,72,152,72,169,42,162,192,160,92,141,10,212,141,24,208
30 DATA 142,23,208,140,26,208,104,168,104,170,104,64
35 POKE 512,0:POKE 513,6
40 POKE 54286,192
45 POKE 752,1:REM TURN OFF CURSOR
50 POSITION 6,11:? #6;"DISPLAY LIST INTERRUPT"
55 POSITION 5,12:? #6;"WITH THREE COLOR CHANGES"
60 POSITION 0,0:POKE 752,0:REM RESTORE CURSOR
```

DEVELOPING A CUSTOM DISPLAY LIST

STEP 1

Make a rough sketch of what you want to appear on the screen. Our example appears as Figure 1.

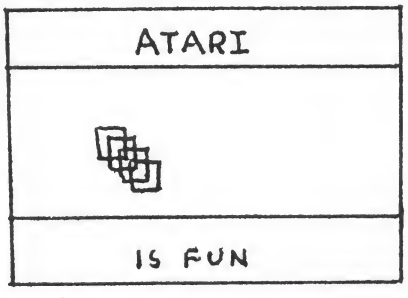

Figure 1

STEP 2

Select the Graphics Modes you want to use and the number of lines for each mode. Two requirements must be met. First, the total number of scan lines in all the mode lines should not exceed 192. If it does, the screen image may "roll." However, the total can be less than 192 with no adverse effect. Second, when you insert new mode lines into an existing display list, the total number of bytes required for the inserted lines must be a whole multiple of the

bytes required per mode line in the existing display list. To understand this more fully, refer to Figure 2. Diagrams such as this are invaluable in planning a display list.

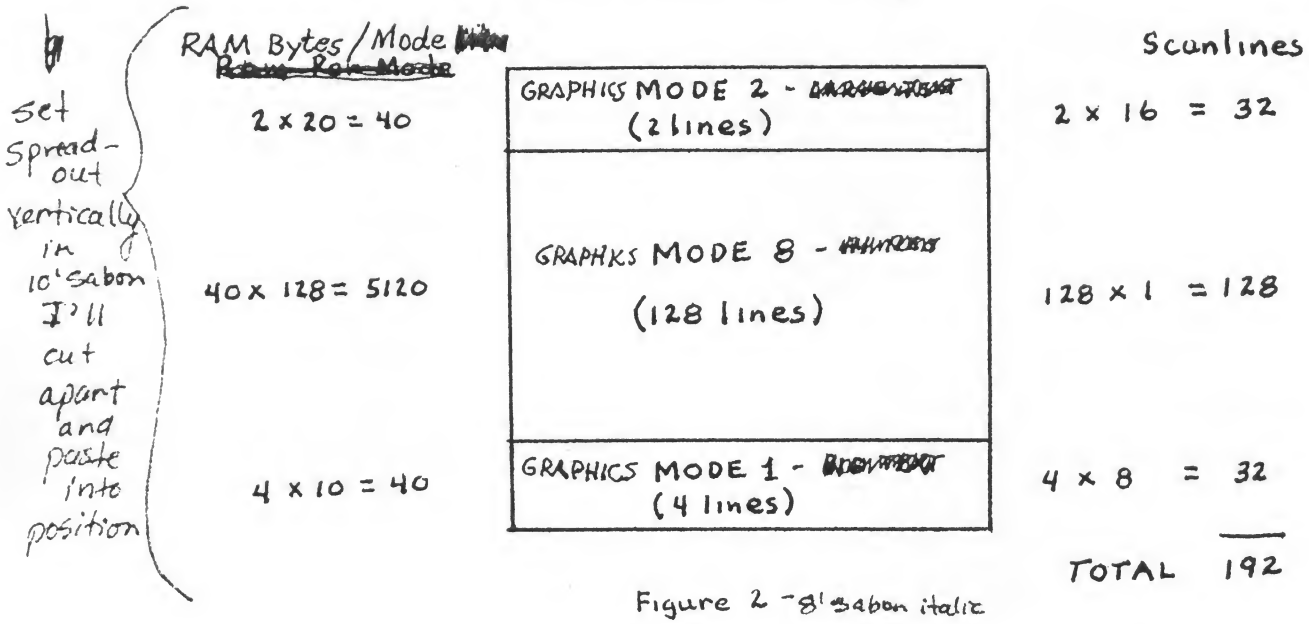

Figure 2 - 8' Sabon italic

Our example will modify a Graphics 8 display list. Each line of Graphics 8 requires 40 bytes of RAM. Therefore, at the top we must insert at least two lines of Mode 2 (two lines x 20 bytes) to match the 40 bytes per line of Mode 8. At the bottom we will insert four lines of Mode 1, each requiring ten bytes, for a total of 40 bytes.

Matching up the byte requirements between inserted lines and existing lines insures that the ~~GR. and ANTIC in~~ ~~synchronized~~ text and graphics will appear where we want them.

Bold QC

STEP 3 After chosing the modes you want, ~~Always start with the Mode~~ determine from Table 1 which of them ~~the modes~~ ~~you have~~ ~~chosen~~ requires the most RAM. Use this mode as your base (existing) mode, ~~maximum amount of~~ RAM. ~~We will modify~~ onto which you make changes that ~~the display list for the mode to~~ create your custom display list. This ~~Using the maximum RAM mode~~ insures that the OS has set aside sufficient memory to hold your screen data.

Mode 8 requires the most RAM, so it will be our base mode, called in line 30, but

~~We are now ready to write a display list program. For our example~~ We have chosen Modes 2,8 and 1. First we'll write a line to clear the screen and turn off the cursor:

```
20 ? CHR$(125):POKE 752,1
```

Next we call the display list to be modified:

```
30 GRAPHICS 8+16
```

Typesetter

et program lines indented with line numbers alligned.

~~We recommend~~ Adding 16 to GR.8 ~~creates~~ eliminates the ~~extra~~ GR.0 window that is a normal part of GR.8. ~~and gives you a full screen to work with~~.

We recommend that you Enter the program as we go along. It will help you understand the process.

STEP 4

PEEK the display list pointer and assign it to a variable such as, "DL".

```
40 DL=PEEK(560)+PEEK(561)*256+4
```

The number 4 is added to the display list pointer for insurance. Recall that the TV generates scan lines that do not appear on the screen. To allow for this, BASIC Graphics Modes generate 24 blank scan lines at the start of the display list. Adding 4 to the display list pointer will make sure that we don't indavertantly remove any of these lines.

STEP 5

POKE the LMS instruction into DL-1. The value 71 derives from ANTIC mode number 7, plus 64. This instruction will establish the first mode line of the display list. If your first mode line belongs to your base mode, skip this step:

```
50 POKE DL-1,71
```

STEP 6

Every mode line in your diagram requires a statement in your

display list. Write these in the same order as they appear on the screen, and POKE the ANTIC Mode numbers as appropriate.

```
60 POKE DL+2,7
```

This is the second line of our Graphics Mode 2.

From the diagram we can see that the next 128 lines are Graphics 8. Since this is our base mode, these lines already exist in the display list. The next mode lines to insert are the four Graphics 1 lines, at the bottom.

```
 70 POKE DL+132,6
 80 POKE DL+133,6
 90 POKE DL+134,6
100 POKE DL+135,6
```

STEP 7

End the display list with a JVB, followed by the low byte and high byte of the return address:

```
110 POKE DL+136,65
120 POKE DL+137,PEEK(560)
130 POKE DL+138,PEEK(561)
    140 GOTO 140
```

Now RUN the program. You will see the top section (GR.2) black, the bottom section (GR.1) black, and the middle section (GR.8) blue. To make the middle section black, change line 30 to:

```
30 GRAPHICS 8+16: SETCOLOR 2,0,0
```

Table 3 shows the relevant portions of our display list and demonstrates another important point. Line 30 of our program has stored the LMS instruction in Address 32851. Line 40 stores the value 7 in Address 32854 to give us the second mode line of Graphics 2. Instructions for the Graphics 1 line and JVB are stored in Addresses 32986 through 32992.¶Look at Addresses 32947 through 32949. Note that here in the middle of the display list is another LMS instruction followed by a screen memory address! The reason is that ANTIC cannot address a block of memory longer than 4K bytes. Since Graphics 8 requires 8K bytes, the screen memory must be broken up into two blocks. ANTIC is sent to the first block of screen memory by the first LMS instruction in Address 32851, and is sent to the second block of screen memory by the second LMS instruction in Address 32947. "Jumping the 4K boundary," occurs only for Graphics 8. You must be careful of two things when you modify a Graphics 8 display list.

First, don't clobber the second LMS instruction and the two bytes following by putting 4 mode lines in their place. Second, you must calculate an offset if you change modes after the boundary jump, ~~near the bottom of the screen.~~ We did this in line 70! by adding two lines to the display list (DL+132 vs. DL+130). ~~in line 50. [illegible] the 4K boundary jump.~~

We urge you to type in lines 20 to 140. In this way you can see the screen display and the process will be more meaningful.

At this point the actual display is written into screen memory. The NEXT task will be to print "ATARI" in the Graphics 2 section. ~~When the graphics statement was entered~~ Line 10 established GR.8 and ~~was automatically~~ instructed the OS that data in screen memory is to be interpreted as graphics, not text. Consequently if we simply enter PRINT #6; "ATARI", the OS will not carry out the command. The OS must be told how to interpret the data it finds in screen memory by POKEing the appropriate Graphics Mode number into memory address 87.

```
140 POKE 87,2
```

```
150 POSITION 8,0:PRINT #6; "ATARI"
```

The OS positions text or graphics on the screen by counting bytes from the start of the screen memory associated with the Graphics Mode value stored in location 87. Thus, it is possible for total screen memory to be considerably longer than the memory for the mode the OS is using. This disparity can cause "cursor out of range" error messages and trouble positioning material on the screen. The cure for both problems is fairly simple. Before creating a display on the screen, change the start of screen memory to coincide with the start of the mode section where you want the display to appear. For the Graphics Mode 8 section this will eliminate the trial-and-error method of placement. For the Graphics Mode 1 section this will prevent a "cursor out of range" message.

To write our display we start with:

```
160 POKE 87,8
```

to tell the OS what mode we're in. Then locate the current top of the screen address with:

```
170 TPSCRN=PEEK(88)+PEEK(89)*256
```

Next, offset the variable TPSCRN by the number of bytes in the Mode 2 lines + 1 (four Mode 2 lines x ten bytes per line = 40 bytes):

```
180 TPSCRN=TPSCRN+41
```

Finally, POKE this memory location back into 88 (low byte) and 89 (high byte):

```
190 POKE 88,TPSCRN-(INT(TPSCRN/256)*256)

200 POKE 89,INT(TPSCRN/256)
```

This procedure sets up the Graphics 8 section of our display so that the top left hand corner corresponds to position 0,0. You can appreciate how much simpler it will be to place your display components.

```
210 COLOR 1:FOR I=1 TO 40 STEP 5

220 PLOT 60+I,40+I:DRAWTO100+I,40+I:DRAWTO 100+I,80+I:
    DRAWTO 60+I,80+I:DRAWTO 60+I,40+I

230 NEXT I
```

Finally, print "IS FUN" in the Mode 1 section at the bottom of the screen.

```
240 POKE 87,1

250 TPSCRN=TPSCRN+5121
```

Line 250 offsets TPSCRN to the beginning of the Mode 1 section. 5121 is obtained from (128 lines of Gr. 8) * (40 bytes per line) = (5120 bytes)+1.

```
260 POKE 88,TPSCRN-(INT(TPSCRN/256)*256)

270 POKE 89,INT(TPSCRN/256)

280 POSITION 6,2:?#6; "IS FUN"

290 GOTO 290
```